

REMARKS/ARGUMENTS

Applicants thank the Examiner for their careful review of this application. Claims 1, 3, 4, 6-8 and 10-20 have been rejected. Claims 1, 6-8, 13-15, and 17-19 have been amended to further clarify that “state” as recited in the claimed invention is “associated with one or more fields defined by an abstract schema” where the abstract schema is “capable of being mapped to a physical schema.” Support for these amendments can be found, among other places, in the Application at page 4, lines 5-8; page 8, lines 9-10; page 15, lines 16-18. It is respectfully submitted that the pending claims define allowable subject matter. Applicants respectfully request reconsideration of the application in view of the above amendments and the following remarks submitted in support thereof.

Discussion of Objection to the Specification

In Section 5, of the Office Action, the Examiner objected to the specification because of the following informality: on page 4, line 18, the phrase “the order of five 9’s” should be explained in more detail. The Examiner requested appropriate correction.

On September 28, 2006, Applicants explained to the Examiner in a telephonic interview that the phrase “the order of five 9’s” is a widely known expression of art, especially as it relates to carrier grade applications, meaning “% 99.999.” The Examiner requested that Applicants provide this explanation in the remarks of this paper. The Examiner further stated that no amendment to the specification would be required.

Applicants have complied with the Examiner’s request and believe that the objection is overcome.

Discussion of Rejection of Claims 1, 3-4, 6-8 and 10-18 under 35 U.S.C. § 102(e)

In Section 7 of the Office Action, the Examiner rejected Claims 1, 3-4, 6-8 and 10-18 under 35 U.S.C. § 102(e) as being anticipated by U.S. Pat. No. 6,298,478 to Nally et al.

The Examiner's rejection is respectfully traversed. First, in the Office Action, the Examiner asserts that Nally et al. discloses a method for upgrading managed state for a JAVA based application wherein an upgraded state object is generated by *upgrading a physical schema using data stored in a repository that is part of a database*, as recited in independent Claim 1. Moreover, in a telephonic interview with the Examiner and the Examiner's Supervisor on October 6, 2006, Applicants respectfully requested that the Examiner identify where the Nally et al. reference either implicitly or explicitly discusses a *physical schema*. In response, the Examiner stated that the transaction tree 400 illustrated in FIG. 4 is the same as a *physical schema*. Applicants respectfully disagree. Nally et al. describes the transaction tree 400 of FIG. 4 as:

"a transaction tree 400 with which multiple nested transactions may be managed A shared transaction 410 is provided at the top of the tree. ... This shared transaction 410 is shown in FIG. 4 as being the parent transaction of two child transactions 420 and 470. ... Each transaction and subtransaction has a view^[1] logically associated therewith. For example, the top-level transaction 410 has a view 415 associated therewith; the subtransaction 420 has a view 425 associated therewith, etc. ... The tree structure depicted in FIG. 4 was used in the related application for managing transactions on objects, and is used in a similar manner in the present invention for managing transactions on EJBs." See Nally et al. at column 10:46 – 11:7 (*emphasis added*).

¹ "In this use of the 'view' concept, the current representation of information the application sees in a view is the current representation of the EJB. Thus by providing each transaction, and each subtransaction (when appropriate), its own view of the EJB, each transaction and subtransaction is able to see a completely independent representation of the EJB that is isolated from any other transaction and subtransaction." See Nally et al. at col. 12, lines 29-36. "It is important to note that the data represented in one subtransaction's view for a particular EJB may be different from the data represented in another subtransaction's view within the same transaction." See Nally et al. at col. 12, lines 63-66.

In other words, the transaction tree 400 of Nally et al. is a tree-like data structure that is used by an application **for managing multiple nested transactions on EJBs**. Conversely, as recognized by those of ordinary skill in the art, a *physical schema* defines the structure of a database, e.g. its tables and associated columns etc., and the type of contents that each data element within the structure contains. And, as further clarified by the specification of the Application, the *physical schema* used by the underlying database is mapped to an *abstract schema*² of a set of entity bean classes. See Application at page 18, lines 1-2; see also EJB 2.0 Specification, Sun Microsystems Inc. at page 127, section 10.2 (EJB 2.0 Specification is incorporated by reference in the Application at page 3, lines 6-10). Specifically, persistent *fields* and *relationships* defined by the *abstract schema* are mapped to the database or other persistent store. See EJB 2.0 Specification at page 126, section 10.1; page 156, section 10.3.11; see also Application at page 18, lines 1-2. Based on the discussion above, Applicants are unclear how a tree-like structure used by an application at runtime to manage multiple concurrent transactions on the same EJB object as taught by Nally et al. is the same as the *physical schema* of the claimed invention.

Second, in the Office Action, the Examiner asserts that Nally et al. discloses a method for upgrading managed state for a JAVA based application wherein an original state object is *upgraded by generating an upgraded state object using upgraded class files*, as recited in independent Claim 8. Applicants respectfully disagree. Specifically, the Examiner asserts that transaction tree 400 illustrated in FIG. 4 and column 14, lines 2-12 of Nally et al. teach using upgraded *class files* to generate an *updated state object*. Applicant's have reviewed the transaction tree 400 illustrated in FIG. 4 and re-read column 14, lines 2-12, and nowhere do

² An *abstract schema*, defines an entity bean's *persistent fields* and *relationships*. See Application at page 4, lines 5-8.

Nally et al. teach or suggest *class files* or upgrading state objects using *class files*. In fact, nowhere are *class files* mentioned. As discussed above, FIG. 4 merely illustrates a tree-like structure for managing multiple concurrent transactions on an EJB object. And column 14, lines 2-12 of Nally et al. reads as follows:

“if top-level transaction 420 from FIG. 4 refers to an EJB 500 for the first time, its view 425 will then have an instantiation of the structure for that EJB as depicted by elements 520, 530, 540, 550, in FIG. 5. (The EJB Object 510 for an application is created the first time any transaction in the application refers to the EJB 500). Suppose that subtransaction 400 then accesses the same EJB 500 to make changes to it. The access itself will cause a version 521 to be copied from the view at 425 into the views at 435 and 445. The subtransaction 440’s changes will then modify the view at 445.”

Applicants respectfully point out that as recognized by those of ordinary skill and as further clarified by the specification of the Application, in a Java-based application a *class* is a structure that defines data and the methods that operate on the data. See Application at page 4, lines 3-5; page 23, line 20 – page 24, line 3; FIG. 6; see also EJB 2.0 Specification at page 190, section 10.6.1; page 158 (illustrative only). And as further clarified in the specification of the Application, an *original state object* is upgraded using upgraded *class files*. See Application at page 14, lines 4-8; page 25, lines 16-21; page 28, lines 12-23. Therefore, because Applicants cannot find where in Nally et al. *class files* are discussed or used for upgrading a state object, Applicants respectfully request the Examiner to specifically identify those portions of FIG. 4 and Column 14, lines 2-12 where a discussion of a *class* or *class files* is found.

Third, in the Office Action, the Examiner asserts that Nally et al. discloses a method for upgrading a JAVA based application having a managed application state where *an original state object stores a state of an original entity bean*, as recited in independent Claim 15. Applicants again respectfully disagree. The Examiner cites to column 14, lines 21-39 of

Nally et al. in support of its assertion. Column 14, lines 21-39 of Nally et al. reads as follows:

“The version status 530 includes information as to whether the EJB has been newly created by an application (‘state=new’) or was read from a database or data store (‘state=old’), and whether the version has been marked for deletion (‘delete-[yes\no]’). If the version is marked as ‘delete=yes’, this status is promoted upward throughout the transaction tree upon a commit so that a delete command can be issued to the persistent store when the top-level transaction commits. The version status data 530 also includes a ‘modified’ flag that is used to synchronize the versions internally (within the transaction tree), and an ‘external-synchronization’ flag that indicates whether the version needs to be synchronized with the persistent store. This external-synchronization flag is set to ‘yes’ whenever an entity bean has been changed. Also, a modification level or number is maintained in each version. The modification level of a version in a child is compared with the modification level in its parent when determining whether a merge will be successful.”

Nowhere in the citation provided by the Examiner do Nally et al. implicitly or explicitly discuss an *original state object* storing a *state* of an original entity bean, as recited in the claimed invention. See Application at page 4, lines 5-8 (“a *field* or *relationship* in the persistent *state* of [an] entity bean”)(*emphasis added*); page 8, lines 9-10; page 15, lines 16-18; page 16, lines 19-20. Rather, column 14, lines 21-39 discusses an approach for committing multiple simultaneous changes to the values of instance data for particular versions of an entity bean. For example, column 14, lines 51-61 of Nally et al. reads as follows:

“The instance data 550 includes the current values for instance data for this version of the entity bean. The application may be referencing the instance values (in read-only mode), or it may be changing them (in a read-write mode). When the instance data 550 for this version 520 of an entity bean has not been changed, the version status data 530 will include a ‘modified=no’ flag; otherwise, the flag will be set to ‘modified=yes’. (While values of flags are described here using text, it will be obvious to one of ordinary skill in the art that an implementation will preferably treat these values as binary data.”

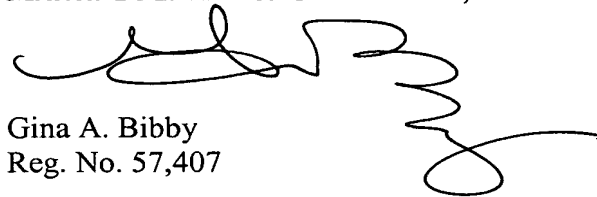
Applicants respectfully note that instance data values depend on the persistent *state* of an entity bean; instance data values do not constitute the persistent *state* of an entity bean. *See e.g., EJB Specification 2.0* at page 155, section 10.3.9.

The foregoing demonstrates that the cited primary reference of Nally et al. do not teach each and every element of independent Claims 1, 8, and 15. Consequently, Nally et al. do not anticipate independent Claims 1, 8, and 15. Additionally, as Claims 3-4, 6-7, 10-14, and 16-20 respectively depend from allowable base Claims 1, 8, and 15; they too are not anticipated by Nally et al. Accordingly, Applicants submit that Claims 1, 3-4, 6-8, and 10-20 are patentable under 35 U.S.C. § 102(e) over Nally et al. Applicants therefore respectfully request reconsideration, and withdrawal of the § 102 rejections.

Conclusion

In view of the foregoing, the Applicant respectfully submits that all the pending Claims 1, 3, 4, 6-8 and 10-20 are in condition for allowance. Accordingly, a Notice of Allowance is respectfully requested. If the Examiner has any questions concerning the present amendment, the Examiner is requested to contact the undersigned at (408) 749-6920. If any additional fees are due in connection with filing this amendment, the Commissioner is also authorized to charge Deposit Account No. 50-0805 (Order No. SUNMP007). A duplicate copy of the transmittal is enclosed for this purpose.

Respectfully submitted,
MARTINE PENILLA & GENCARELLA, L.L.P.



Gina A. Bibby
Reg. No. 57,407

Martine Penilla & Gencarella, LLP
710 Lakeway Drive, Suite 200
Sunnyvale, California 94085
Telephone: (408) 774-6920
Customer Number 32291